Final Report on

Development of a Newton-Krylov Iterative Method to Address Strong Non-Linear Feedback Effect in BWR Core Simulators

Award Number: DE-FG07-99ID13773

by

Doddy Kastanya and Paul J. Turinsky

Electric Power Research Center
Department of Nuclear Engineering
Box 7909
North Carolina State University
Raleigh, NC 27695-7909

November 7, 2002

Abstract

A Newton-BICGSTAB solver has been developed to reduce the CPU execution time of BWR core simulators. The new solver treats the strong non-linearities in the problem explicitly using the Newton's method, replacing the traditionally used nested iterative approach. The Newton's method provides the solver with a higher-than-linear convergence rate, assuming that a good initial estimate of the unknowns is provided. Within each Newton iteration, an appropriately preconditioned BICGSTAB method is utilized for solving the linearized system of equations. Taking advantage of the higher convergence rate provided by the Newton's method and utilizing an efficient preconditioned BICGSTAB solver, we have developed a computationally efficient Newton-BICGSTAB solver to evaluate the three-dimensional, two-group neutron diffusion equations coupled with a two-phase flow model within a BWR core simulator.

The robustness of the solver has been tested against numerous BWR core configurations and consistent results have been observed each time. The Newton-BICGSTAB solver provides an overall speedup of around 1.7 to the core simulator, with reference to the traditional approach. Isolating the solver portion of the core simulator, one can see that the new algorithm actually provides a speedup of around 1.9, of which 48% can be attributed to the BICGSTAB solver and the remaining 52% to Newton's method.

Table of Contents

I. INTRODUCTION

In the course of performing a boiling water reactor (BWR) reload core analysis, a core simulator is extensively used in the preliminary process of choosing a pair of loading pattern-control rod pattern (LP-CRP) by verifying the attractiveness of the pair for various normal operating conditions as well as different accident scenarios to guarantee a safe and economical operation of the reactor. Therefore, having a fast core simulator is an important factor in expediting the reload core design and analysis.

Automated mathematical optimization tools developed for determining an optimum LP-CRP pairing in a BWR core are becoming available and can be used by a core designer to ease the process of selecting a good LP-CRP pair for a particular reload core design. Independent of the optimization method employed by the tools, *e.g.* simulated annealing (SA), genetic algorithm (GA), neural network or taboo search, these tools have to evaluate thousands of LP-CRP pairings to assess their attractiveness before arriving at the optimum pair. Each evaluation involves solving the three-dimensional, two-group neutron diffusion equation coupled with the two-phase fluid flow model, *e.g.* the three-equation mixture drift flux model. Given that these tightly coupled non-linear equations must also be solved as a function of cycle burnup, the associated computational burden could become prohibitive.

One option to accelerate the optimization process is to tune the optimization parameters of the employed optimization method in order to reach the optimum solution faster and, hence, reducing the number of LP-CRP pairs to evaluate. In addition, we believe that reducing the time associated with the evaluation of a single LP-CRP pair is a key factor for reducing the overall optimization running time. In this project, this target is pursued by assessing the effectiveness of a

Newton-Krylov solver in replacing the nested iterative approach or similar methods currently utilized by most BWR core simulators for treating the non-linear feedback properties. Addressing the non-linearity of the problem explicitly using the Newton's method, which is capable of performing simultaneous updates of all unknowns, aided by an appropriately preconditioned Krylov method for solving the linearized system of equations within the Newton iteration, should reduce the CPU execution time of an optimization run.

This paper summarizes the development of a robust and computationally efficient Newton-Krylov algorithm to solve the three-dimensional, two-group neutron diffusion equations coupled with a two-phase flow model. To assess the performance of this new algorithm, we have implemented the Newton-Krylov algorithm in the core simulator part of the large-scale in-core fuel management optimization code FORMOSA-B [7, 10] replacing the currently utilized traditional nested iterative algorithm. Although results reported in this paper are based upon the implementation in the FORMOSA-B code, they should be reproducible by other BWR core simulators.

This paper highlights the important steps in the development of the Newton-Krylov solver. Section II discusses the basic questions need to be answered in developing the new solver. Mathematical background on the development of the solver is also briefly presented in this section along with the generic form of the Newton-Krylov algorithm. The efforts in developing an appropriate preconditioner to accompany the selected Krylov solver are discussed in Section III. Section IV explores the possibility of implementing an inexact Newton's method. This section only introduces the variants of inexact Newton's method examined, with the performance of these methods discussed later in the paper. Section V presents some results from testing the new solver on a single loading pattern, *i.e.* performing a load follow calculation, as well as on multiple LP-

CRP pairs, *i.e.* performing an optimization run. Finally, Section VI presents conclusions and provides suggestions for future research to improve the performance of the new solver.

II. METHODOLOGY

II. A. Basic Questions to be Addressed

In solving the neutron diffusion equation, non-linearities originate through neutron cross section dependencies on thermal hydraulics conditions, *e.g.* fuel temperature, coolant density, coolant temperature, and transient fission product number densities, *e.g.* Xe^{135} and Sm^{149}. In addition, the nature of the iterative mathematical solver may also introduce non-linearities, *e.g.* non-linear nodal iterative method. Given this large number of sources of non-linearity, several questions related to the implementation of the Newton-Krylov solver have had to be addressed.

The first of many basic questions to be addressed is related to determining which unknowns, including the non-linear feedbacks unknowns, should be updated simultaneously by the Newton's method. To answer this question, we have performed a comprehensive study to determine which feedback mechanisms can be categorized as strong feedbacks. If the total number of outer (fission-source) iterations increases noticeably when a non-linear feedback effect is treated, then the feedback is categorized as a strong feedback. The non-linear feedbacks associated with the thermal hydraulics, coolant inlet flow redistribution, transient fission products, and the non-linear nodal iterative method spatial coupling correction coefficients have been examined. The behaviors of the non-linear feedbacks on four core types, a 368 assembly GE BWR/4 core, an 800 assembly GE BWR/6 core, a 724 assembly GE BWR/3 core, and a 560 assembly GE BWR/4 core, have been evaluated to verify the consistency of the results.

Table I presents the cases examined during this study and the stopping criteria utilized. The results of this study are summarized in Table II. Included in this table are results from performing the study using two different sets of stopping criteria. The cases examined are chosen such that the individual effect of a certain feedback can be isolated (cases 1, 2, 3, and 5) as well as the aggregate effect of several feedbacks (cases 4 and 6 through 10). Note that the ratio is defined as

$$Ratio = \frac{n_{outer}^{Case}}{n_{outer}^{Ref}}, \tag{1}$$

where n_{outer}^{Case} denotes the number of outer iterations for a particular combination of non-linear feedbacks being tested and n_{outer}^{Ref} denotes the number of outer iterations without nodal, thermal hydraulics, coolant inlet flow redistribution, and critical flow search feedbacks treated.

Averaged over the four core types, Case 1 shows that utilization of a nodal method in place of the coarse mesh finite difference (CMFD) method increases the number of outer iterations on average by 12% and 19% for the loose and tighter stopping criteria, respectively. Due to these moderate changes in the number of iterations, this feedback mechanism is categorized as a weak feedback.

Case 2 shows the effect of thermal hydraulics feedback without flow redistribution, critical flow search, and spacer void model. Averaged over four core types, this feedback effect is considered a strong feedback mechanism since it increases the number of outer iterations by around 50%, regardless of the stopping criteria employed.

Comparing the ratio values for Cases 2 and 3, one can see that performing the inlet flow redistribution, such that the core has uniform inlet and outlet pressures for all flow channels, does

not significantly change the number of outer iterations. Similar observation can also be made on Cases 6 and 7, where the nodal method is utilized for solving the three-dimensional, two-group neutron diffusion equation instead of the CMFD method. Therefore, the flow redistribution feedback is categorized as a weak feedback.

Observing the ratio values of Cases 3 and 4 as well as Cases 7 and 8, one can see that the critical flow search feedback mechanism can actually be categorized as a strong feedback. This feedback adds 9% to 83% increase in the number of outer iterations on top of the already strong thermal hydraulics feedbacks. However, in this project the critical flow search feedback will be treated in a nested iterative fashion with other weak feedbacks.

Although employing the spacer void model does not change the number of outer iterations considerably, it will still be treated using the Newton's method since the spacer void model directly affects the coolant density distribution which is a part of the thermal hydraulics model. Comparing the ratio values of Cases 10 and 8, with an exception for the 724 assembly GE BWR/ 3 core, one can see that the fission product feedback changes the number of outer iterations by less than 10% and, hence, is categorized as a weak feedback.

Based on these results, we concluded that the thermal hydraulics feedbacks are considered strong feedbacks and, therefore, their associated unknowns will be treated using the Newton's method. The remaining weak feedbacks will be treated in a nested iterative fashion.

Since a nested iterative approach is to be employed, we need to know what convergence or stopping criteria should be utilized to guarantee the minimization of total computational costs while still assuring the robustness of the solver. Discussions about the algorithm for the nested iterative approach and the associated stopping criteria are presented later in this section.

A linearized system of equations needs to be solved during each Newton iteration. We need to decide on what solver to employ for solving these sets of linearized system of equations. We decided to use one of the Krylov subspace solvers to perform this task. Our decision is primarily influenced by the earlier success of utilizing the preconditioned BICGSTAB solver in providing the NESTLE code [13] with substantial speedups, with reference to the red/black line successive over relaxation (R/B LSOR) solver. The question about which Krylov solver to utilize and what preconditioner to use will be addressed in this paper.

II. B. Mathematical Background

Consider a non-linear equation given by

$$\bar{A}(\bar{\Phi}) = \bar{Q}, \tag{2}$$

where the non-linear operator $\bar{A}$ operates on the unknown vector $\bar{\Phi}$ whose vector dependence indicates different reactor core properties. Eq. (2) includes appropriate boundary conditions and, if an eigenvalue problem is involved, normalization constraint. For the problem at hand, the unknown vector $\bar{\Phi}$ consists of all the unknowns and can be subdivided into the eigenvalue (λ), where $\lambda = 1/k_{eff}$, the two-group neutron flux ($\bar{\phi}$), the strong non-linear feedback properties ($\bar{\xi}$), *i.e.* fuel temperature, coolant temperature, and coolant density, and the weak non-linear feedback properties ($\bar{\Psi}$), *i.e.* Xe^{135} and Sm^{149} number densities and nodal spatial coupling correction coefficients. The unknown vector can be mathematically written as

$$\bar{\Phi} = [\bar{\phi}^T, \bar{\xi}^T, \lambda, \bar{\Psi}^T]^T. \tag{3}$$

The subdivision on the non-linear feedback properties is performed in order to allow different iterative update frequencies between the strong and the weak feedback properties. Expressing Eq. (2) explicitly in terms of the eigenvalue, neutron flux and feedback properties, we obtain

$$\bar{A}(\bar{\Phi}) = \begin{bmatrix} (\bar{\bar{M}}(\bar{\xi}, \bar{\Psi}) - \lambda \bar{\bar{F}}(\bar{\xi}, \bar{\Psi}))\bar{\phi} \\ \bar{\hat{T}}(\bar{\phi}, \bar{\xi}, \bar{\Psi}) \\ \bar{W}(\bar{\phi}, \bar{\xi}, \bar{\Psi}) \end{bmatrix} = \begin{bmatrix} \bar{0} \\ \bar{\hat{Q}}_{\xi} \\ \bar{0} \end{bmatrix}. \tag{4}$$

The first row of Eq. (4) indicates the balance equation for the neutron flux, *i.e.* the spatially discretized form of the three-dimensional, two-group neutron diffusion equation. $\bar{\bar{M}}$ and $\bar{\bar{F}}$ denote the loss and production operators, respectively. Note that these operators are independent of flux. The second row of Eq. (4) represents some form of two-phase fluid mass, energy, and momentum conservation equations, along with the constitutive equations which are used to determine the coolant density distribution in the core, plus the core power level normalization constraint. The third row states the balance equations for the weak non-linear feedback properties.

The traditional nested iterative algorithm for solving Eq. (4) can be mathematically written as

$$\bar{\bar{M}}(\bar{\xi}_n, \bar{\Psi}_o)\bar{\phi}_{m+1} - \lambda_m \bar{\bar{F}}(\bar{\xi}_n, \bar{\Psi}_o)\bar{\phi}_m = \bar{0}, \text{ for } m = 0, 1, 2, \ldots, \tag{5}$$

$$\bar{\hat{T}}(\bar{\phi}_{\tilde{m}}, \bar{\xi}_{n+1}, \bar{\Psi}_o) = \bar{Q}_{\xi}, \text{ for } n = 0, 1, 2, \ldots, \tag{6}$$

and

$$\bar{W}(\bar{\phi}_m, \bar{\xi}_n, \bar{\Psi}_{o+1}) = \bar{0}, \text{ for } o = 0, 1, 2, \ldots, \tag{7}$$

where m, n, and o denote the fission source, strong nonlinear feedback properties, and weak nonlinear feedback properties iterative counts, respectively. The indices with tildes on top of or

underneath them indicate the frequency of update; for example, several fission source iterations could be performed prior to updating the strong nonlinear feedback properties.

Ideally, it is desired to update the flux and the strong non-linear feedback properties simultaneously. This can be done through a Newton type method [8, 9]. The implementation of Newton's method in this paper is focused only on treating the flux and the strong non-linear feedback properties, *i.e.* the thermal hydraulics feedback properties. The remaining weak non-linear feedback properties will be updated in an iterative fashion as additional loops outside the Newton iteration loop.

The non-linear system of equations we are interested in solving utilizing the Newton's method can be written as

$$\bar{B}(\bar{\Theta}) \equiv \begin{bmatrix} (\bar{\bar{M}}(\bar{\xi}, \bar{\Psi}) - \lambda \bar{\bar{F}}(\bar{\xi}, \bar{\Psi}))\bar{\phi} \\ \bar{T}(\bar{\phi}, \bar{\xi}, \bar{\Psi}) \\ \langle \overline{\kappa \Sigma_f}, \bar{\phi} \rangle \end{bmatrix} = \bar{Q} \equiv \begin{bmatrix} \bar{0} \\ \overline{Q_\xi} \\ Q_P \end{bmatrix}. \tag{8}$$

The core power level constraint equation has been separated out of the $\bar{\hat{T}}$ operator in Eq. (8), with the $\bar{T}$ operator denoting the two-phase flow thermal hydraulics conservation and constitutive equations. As noted earlier, since an eigenvalue equation is being solved, a constraint equation is required, in this case to normalize the flux. The constraint equation allows the eigenvalue to be introduced as an additional unknown. The exact Newton equation for non-linear equation, Eq. (8), suppressing the dependence upon the weak non-linear feedbacks, $\bar{\Psi}$, which are updated in an outer iterative loop, can be expressed as

$$\bar{\bar{J}}(\bar{\Theta}_m)\bar{\Theta}_{m+1} = \bar{R}(\bar{\Theta}_m), \tag{9}$$

where the Jacobian matrix has been defined as

$$\bar{\bar{J}}(\bar{\Theta}_m) \equiv \frac{\partial}{\partial \bar{\Theta}} \bar{B}(\bar{\Theta}) \Big|_{\bar{\Theta}_m}, \tag{10}$$

$$\bar{\Theta}_m \equiv [\bar{\phi}_m^T, \bar{\xi}_m^T, \lambda_m]^T, \tag{11}$$

and

$$\bar{R}(\bar{\Theta}_m) \equiv \bar{\bar{J}}(\bar{\Theta}_m)\bar{\Theta}_m - (\bar{B}(\bar{\Theta}_m) - \bar{Q}), \tag{12}$$

with index m now denoting the Newton iteration count. For our problem, the Jacobian matrix can be written as

$$\bar{\bar{J}}(\bar{\Theta}_m) = \begin{bmatrix} [\bar{\bar{M}}(\bar{\xi}) - \lambda \bar{\bar{F}}(\bar{\xi})] & \left[\left(\frac{\partial}{\partial \bar{\xi}} \bar{\bar{M}}(\bar{\xi}) - \lambda \frac{\partial}{\partial \bar{\xi}} \bar{\bar{F}}(\bar{\xi})\right)\bar{\phi}\right] & -\bar{\bar{F}}(\bar{\xi})\bar{\phi} \\ \frac{\partial}{\partial \bar{\phi}} \bar{T}(\bar{\phi}, \bar{\xi}) & \frac{\partial}{\partial \bar{\xi}} \bar{T}(\bar{\phi}, \bar{\xi}) & \bar{0} \\ \frac{\partial}{\partial \bar{\phi}} \langle \overline{\kappa \Sigma_f}, \bar{\phi} \rangle & \frac{\partial}{\partial \bar{\xi}} \langle \overline{\kappa \Sigma_f}, \bar{\phi} \rangle & 0 \end{bmatrix} \Bigg|_{\bar{\Theta}_m}. \tag{13}$$

Eq. (13) denotes a 3x3 block matrix whose block matrix components are denoted by $\bar{\bar{J}}(\bar{\Theta}_m)_{i,j}$ for $i, j = 1, 2, 3$. Using Eqs. (9) and (13), one obtains the following iterative algorithm (in similar fashion to Eqs. (5) through (7)):

$$\begin{aligned} &\bar{\bar{J}}(\bar{\Theta}_m)_{1,1}\bar{\phi}_{m+1} + \bar{\bar{J}}(\bar{\Theta}_m)_{1,2}\bar{\xi}_{m+1} + \bar{\bar{J}}(\bar{\Theta}_m)_{1,3}\lambda_{m+1} = \\ &(\bar{\bar{J}}(\bar{\Theta}_m)_{1,1}\bar{\phi}_m + \bar{\bar{J}}(\bar{\Theta}_m)_{1,2}\bar{\xi}_m + \bar{\bar{J}}(\bar{\Theta}_m)_{1,3}\lambda_m) - \\ &[\bar{\bar{M}}(\bar{\xi}_m) - \lambda_m \bar{\bar{F}}(\bar{\xi}_m)]\bar{\phi}_m \end{aligned}, \tag{14}$$

$$\begin{aligned} &\bar{\bar{J}}(\bar{\Theta}_m)_{2,1}\bar{\phi}_{m+1} + \bar{\bar{J}}(\bar{\Theta}_m)_{2,2}\bar{\xi}_{m+1} = \\ &\bar{\bar{J}}(\bar{\Theta}_m)_{2,1}\bar{\phi}_m + \bar{\bar{J}}(\bar{\Theta}_m)_{2,2}\bar{\xi}_m - [\bar{T}(\bar{\phi}_m, \bar{\xi}_m) - \overline{Q_\xi}] \end{aligned}, \tag{15}$$

and

$$\bar{\bar{J}}(\bar{\Theta}_m)_{3,1}\bar{\phi}_{m+1} + \bar{\bar{J}}(\bar{\Theta}_m)_{3,2}\bar{\xi}_{m+1} =$$

$$\bar{\bar{J}}(\bar{\Theta}_m)_{3,1}\bar{\phi}_m + \bar{\bar{J}}(\bar{\Theta}_m)_{3,2}\bar{\xi}_m - [\langle(\overline{\kappa\Sigma_f})_m\bar{\phi}_m\rangle - Q_P] \quad . \tag{16}$$

Reintroducing the weak non-linear feedback values, they are determined by solving for $\bar{\Psi}$ using

$$\bar{W}(\bar{\Theta}_m, \Psi_{o+1}) = \bar{0}, \tag{17}$$

This set of equations is clearly more difficult to solve than the original matrix equations associated with the traditional nested iterative approach due to the simultaneous couplings of more unknowns. However, the Newton's method is still attractive since it converges to the solution at a-higher-than-linear rate provided that a good initial guess of unknown values is used.

II. C. Linearization Error Study

In order to assess the feasibility of employing the Newton's method for treating the strong non-linearities in the BWR core simulator model, we would like to first understand how accurate the first order approximation is for our problem. A linearization error study has been performed to answer this question. In addition to observing how significant the errors introduced by using the first-order Newton's method are, this study was performed to also check the singularity property of the Jacobian matrix system at convergence and to serve as a tool to validate the derivation and implementation of the linearized system of equations.

The linearization error study was performed on a 368 assembly GE BWR/4 core. Since this study is focused on a single Newton step, the core simulator options for treating non-linear feedbacks not explicitly treated with the Newton's method, *e.g.* inlet coolant flow redistribution, critical flow search, and fission products, are turned off. The study begins by obtaining the solution at the 100% power level. Then, we performed linearization around this solution, *i.e.* evaluate the Jacobian matrix at 100% power, and subsequently solve for the changes in the

eigenvalue, two-group neutron flux, moderator pressure, density, void fraction, and internal energy distribution at a perturbed core power level, 100±x%, *via* one Newton iteration. We also solve for these core variables at the perturbed power level using a traditional iterative solver. The difference between the exact changes, *i.e.* traditional solver, and the approximate changes, *i.e.* one Newton iteration, in these variables are defined as the linearization errors.

It is anticipated that the linearization errors will grow as the value of the core power level perturbation increases. However, since the two-group neutron flux and the thermal hydraulics properties have spatial dependencies, it is not trivial to succinctly quantify the degradation in the quality of the solutions as a result of applying a larger power perturbation. The error in the eigenvalue as a function of core power level perturbation size is selected to serve as the overall measure of linearization error. This is a compact manner to do the linearization error analysis since the eigenvalue is a unique quantity for each power level and, hence, the accuracy in predicting this value can be easily plotted as a function of core power level.

The comparison between the approximate and the exact change in k_{eff} as well as the error of the predicted change in k_{eff} are presented in the top and bottom plots of Figure 1, respectively. From these graphs, one can see that utilizing one iteration of the first order Newton's method, the change in the eigenvalue can be quite accurately predicted. As expected, the error grows as the core being examined moves further away from the reference 100% core power level. However, one should keep in mind that during the course of Newton iterations, the fluctuations in core power will not likely come close to that caused by the full range of core power level perturbation analyzed in this study. This follows since the Jacobian matrix will be updated at each Newton iteration using the latest iterative values. Figure 2 shows the comparisons of the approximate and exact changes in the fast flux distributions for four radial locations (as shown in the core map

below the plots) when the core power level is perturbed by +10%, *i.e.* 110% core power level. Although these plots represent an extreme perturbation in the core power, the first-order approximation still predicts the axial power distribution reasonably well. Note that slightly larger differences are observed in radial location 04; however, recognizing the size of the relative power, one can conclude that these larger differences are not too important since the absolute differences are still relatively small. Based on these results, it is safe to assume that the errors introduced by the first-order approximation are insignificant.

The other thing learned from this study is that the Jacobian matrix is not singular. The singularity of the Jacobian matrix becomes a concern since the $\overline{\overline{J_{11}}}$ block, defined as $(\overline{\overline{M}}(\bar{\xi}, \overline{\Psi}) - \lambda\overline{\overline{F}}(\bar{\xi}, \overline{\Psi}))$, is singular when the value of λ is the true eigenvalue associated with the generalized eigenvalue problem. This is indeed the case for our linearization error study since we utilize the eigenvalue problem solution (at 100% power level) as our reference. However, in reality additional terms are added to $\overline{\overline{J_{11}}}$ to account for cross-section changes due to fuel temperature changes. This follows since the cross-section is a function of fuel temperature, which in turns is a function of power density, which in turns is a function of flux, and we reduce the cross-section dependence to flux. These additional terms when added to the $\overline{\overline{J_{11}}}$ block prevent this block from being singular.

II. D. Krylov Methods

Krylov methods are categorized as non-stationary iterative methods. For this family of methods, information needed to advance from one iteration to the next keeps changing. This

means that when solving Eq. (9) for a fixed Newton iteration m, allowing us to suppress the m dependence and rewrite Eq. (9) as $\bar{\bar{J}}\bar{\Theta} = \bar{R}$, we can express the nth Krylov iteration of $\bar{\Theta}$ as

$$\bar{\Theta}_{(n)} = \bar{\bar{G}}\bar{\Theta}_{(n-1)} + \bar{c}, \tag{18}$$

where $\bar{\bar{G}}$ is the iterative matrix and $\bar{c}$ is the remainder. When using unpreconditioned Krylov methods, the nth iteration of $\bar{\Theta}$ is an element of

$$\bar{\Theta}_{(0)} + span\left\{\bar{r}_{(0)}, \bar{\bar{J}}\bar{r}_{(0)}, \ldots, \bar{\bar{J}}^{n-1}\bar{r}_{(0)}\right\} \tag{19}$$

which minimizes some norm measuring the distance between $\bar{\Theta}_{(n)}$ and the true solution, $\bar{\Theta}$. In Eq. (19), $\bar{\Theta}_{(0)}$ represents the initial guess, the initial residual ($\bar{r}_{(0)}$) is defined as

$$\bar{r}_{(0)} = \bar{R} - \bar{\bar{J}}\bar{\Theta}_{(0)}, \tag{20}$$

and $span\left\{\bar{r}_{(0)}, \bar{\bar{J}}\bar{r}_{(0)}, \ldots, \bar{\bar{J}}^{n-1}\bar{r}_{(0)}\right\}$ is the nth Krylov subspace. How the coefficients (multipliers) of the bases for a particular Krylov subspace are calculated is what makes a Krylov method unique from the others. A discussion about the initial development of Krylov methods can be found in the literature [3, 12, 14]. Barrett *et. al.* [1] provides a compact but thorough discussion of Krylov methods and also gives an introduction to some families of preconditioner matrices. Examples of applications of Krylov methods in core simulator codes for reactor analysis are available in [2], [4], [5], [6], and [15].

For this project, we require Krylov methods which can solve a non-symmetric coefficient matrix since our Jacobian matrix is non-symmetric. We decided to examine three such Krylov methods, namely the bi-conjugate gradient stabilized (BICGSTAB), conjugate gradient squared

(CGS), and restart generalized minimal residual (GMRES). The most computationally efficient solver will be employed in the final implementation of the Newton-Krylov solver.

II. E. The Newton-Krylov Algorithm

The algorithm for the Newton-Krylov solver implemented in FORMOSA-B is depicted in Figure 3. The Newton iteration performs simultaneous updates of the eigenvalue, two-group neutron flux, and thermal hydraulics properties. Note that fission source iterations have been completely eliminated. This is possible since the singular matrix system that appears in the eigenvalue problems does not appear in Newton's method, *i.e.* Jacobian matrix is non-singular. Although this figure is fairly self-explanatory, there are two items which should be pointed out. First, the weak non-linear feedback properties are being updated at a certain frequency of Newton (outer) iterations. Second, the box labeled "Setup the Jacobian matrix system and perform the LU factorization" is thicker than the other boxes to indicate that these processes are done at a certain frequency depending upon whether an exact or inexact Newton's method is utilized.

Given that the Newton-Krylov algorithm is a nested iterative algorithm, a specific stopping criteria is needed for each level of the nested iterative loops. As the stopping criteria for the Krylov solver, we require the L_2-norm of the reduction in the residual to be less than a specified value

$$\frac{\left\|\bar{r}_{(m,n)}\right\|_2}{\left\|\bar{r}_{(m,0)}\right\|_2} \leq \varepsilon_{Krylov}, \tag{21}$$

where

$$\bar{r}_{(m,n)} = \begin{bmatrix} \bar{R}_{(m)_1} \\ \bar{R}_{(m)_2} \\ R_{(m)_3} \end{bmatrix} - \begin{bmatrix} \bar{\bar{J}}_{11} & \bar{\bar{J}}_{12} & \bar{\bar{J}}_{13} \\ \bar{\bar{J}}_{21} & \bar{\bar{J}}_{22} & \bar{0} \\ \bar{\bar{J}}_{31} & \bar{\bar{J}}_{32} & 0 \end{bmatrix} \begin{bmatrix} \delta\bar{\phi}_{(m,n)} \\ \delta\bar{\xi}_{(m,n)} \\ \delta\lambda_{(m,n)} \end{bmatrix}, \tag{22}$$

and the pair (m, n) denotes the (Newton, Krylov) iteration step counts. Note that when this Krylov iteration stopping criteria is satisfied, say at Krylov iteration $\hat{n}$, we denote $(m) = (m, \hat{n})$. For terminating the Newton iterations, the following stopping criteria is used:

$$\frac{\left\| \bar{\bar{M}}(\bar{\xi}_m, \bar{\Psi}_o)\bar{\phi}_m - \lambda_m \bar{\bar{F}}(\bar{\xi}_m, \bar{\Psi}_o)\bar{\phi}_m \right\|_2}{\left\| \lambda_m \bar{\bar{F}}(\bar{\xi}_m, \bar{\Psi}_o)\bar{\phi}_m \right\|_2} \leq \varepsilon_{Newton}. \tag{23}$$

We realize that this stopping criteria only checks the neutron diffusion equation solution, while the Jacobian matrix system also includes thermal hydraulics properties. Hence, this stopping criteria was formulated by assuming that when the two-group fluxes and eigenvalue converge to sufficient accuracy, the thermal hydraulics properties have already converged to the accuracy desired. We have performed numerical tests to show that this is indeed a valid assumption. Another advantage of utilizing this stopping criteria is related to the fact that the term whose norm is taken and used as the numerator in Eq. (23) is readily available since it is part of the right hand side vector for the subsequent Newton iteration. Therefore, evaluation of the Newton iteration's stopping criteria does not add much to the computational cost.

Similar to the traditional solver, to terminate the overall solver, we check the convergence of the eigenvalue, fission neutron source density, and moderator density, in the outer most loop of the nested loops, *i.e.* the weak non-linear feedback loop. Mathematically, these stopping criteria can be written as

$$\frac{\left|k_{eff}^{(o)} - k_{eff}^{(o-1)}\right|}{k_{eff}^{o}} \leq \varepsilon_k, \tag{24}$$

$$\frac{\left\|[\bar{\bar{F}}\bar{\phi}]^{(o)} - [\bar{\bar{F}}\bar{\phi}]^{(o-1)}\right\|_2}{\left\|[\bar{\bar{F}}\bar{\phi}]^{(o)}\right\|_2} \leq \varepsilon_F, \tag{25}$$

and

$$\frac{\left\|\rho^{(o)} - \rho^{(o-1)}\right\|_2}{\left\|\rho^{(o)}\right\|_2} \leq \varepsilon_\rho, \tag{26}$$

where the superscript o denotes the weak non-linear feedback iterative count.

III. DEVELOPMENT OF PRECONDITIONERS

In general, the rate of convergence of iterative methods depends on the spectral properties of the coefficient matrix. For Krylov methods, the condition number of the coefficient matrix plays an important role in determining the rate of convergence. One may attempt to transform the original linear system into an equivalent system in the sense that it still has the same solution, but it has a more favorable condition number. Such a transformation can be attained by utilizing a preconditioner matrix.

One needs to realize that utilizing a preconditioner in a Krylov method adds some extra computational time both in the initial setup and during the iterations for applying it. However, there is a trade-off between the increasing computational time associated with constructing and applying the preconditioner and the advantage of increased convergence rate. Therefore, the focus of formulating a preconditioner is to minimize the additional incremental computational time while maximizing the convergence rate so as to reduce the overall computational cost.

Taking advantage of the structure of the coefficient matrix is a crucial factor in devising a preconditioner. Another essential factor in the development of an effective preconditioner is an understanding of the physics behind the constitutive equations forming the linear system. For example, in approximating the original coefficient matrix, one can eliminate some elements of the matrix which are known to be not too important in the sense that ignoring them will still produce a solution close to the exact solution.

In formulating the preconditioners described in this section, a block incomplete LU (BILU) factorization approach [5] is utilized as a part of the preconditioners. This factorization was selected to be incorporated in the preconditioners since it is a straight-forward process which requires a predetermined fixed number of operations dictated by the size of the problem and can capture the stronger spatial and/or energy coupling of unknowns. Note that the size of the matrix block to be factorized varies depending upon the arrangement of the unknowns.

III. A. Porsching's Algorithm

The idea for this algorithm is introduced in a paper by Porsching *et. al.* [11] on hydraulic networks. Using this algorithm, the components of the coefficient matrix of a linear system, $\bar{\bar{A}}\bar{x} = \bar{b}$, are rearranged if possible into the following block form

$$\bar{\bar{A}} = \begin{bmatrix} \bar{\bar{A}}_{11} & \bar{\bar{A}}_{12} \\ \bar{\bar{A}}_{21} & 0 \end{bmatrix}, \tag{27}$$

where, despite the matrix notation, $\bar{\bar{A}}_{12}$ and $\bar{\bar{A}}_{21}$ are actually column and row vectors, respectively. In this section, note that matrix notation will always be used to denote the submatrix component even though sometimes the vector notation is more appropriate. The system to be solved can then be written as

$$\begin{bmatrix} \bar{\bar{A}}_{11} & \bar{\bar{A}}_{12} \\ \bar{\bar{A}}_{21} & 0 \end{bmatrix} \begin{bmatrix} \bar{x}_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} \bar{b}_1 \\ b_2 \end{bmatrix}. \tag{28}$$

The algorithm for solving this system of equations is straight-forward and is described in Figure 4. Porsching's algorithm is used as a part of the preconditioners introduced in the next three subsections.

III. B. Preconditioner 1

The first type of preconditioner examined uses a node-wise grouping of the unknowns, where all of the unknowns belonging to a particular node are grouped together. The preconditioner can be symbolically written as

$$\bar{\bar{P}}_1 = \begin{bmatrix} \bar{\bar{\hat{J}}}_{11} & \bar{\bar{\hat{J}}}_{12} \\ \bar{\bar{\hat{J}}}_{21} & 0 \end{bmatrix} \tag{29}$$

where the "hat" indicates a re-ordered Jacobian matrix block from what was introduced earlier.

The $\bar{\bar{\hat{J}}}_{11}$ block has a block seven banded structure which represents the underlying node coupling of the finite difference method applied to the neutron diffusion equation. Recall we are using a nodal method but employing the non-linear iterative nodal method, which gives rise to the finite difference method spatial coupling. Also included in $\bar{\bar{\hat{J}}}_{11}$ are the thermal hydraulics equations. Each entry of the $\bar{\bar{\hat{J}}}_{11}$ block is a 6x6 matrix, while the entries of $\bar{\bar{\hat{J}}}_{12}$ and $\bar{\bar{\hat{J}}}_{21}$ blocks are 6x1 and 1x6 vectors, respectively, associated with the criticality constant.

This preconditioner is formulated based on the fact that the more components of the original Jacobian matrix are included in the preconditioner matrix, the faster the convergence rate

of the Krylov solver will be. In the outer shell of the preconditioner, the Porsching's algorithm is utilized to take advantage of the structure of the matrix depicted in Eq. (29). Within the Porsching's algorithm itself, we use a BILU approximation to determine the action of $\left(\bar{\bar{\hat{J}}}_{11}\right)^{-1}$, which equates to solving for u and v in Figure 4. Note that some vectors, whose values do not change during Krylov iterations, are saved in order to reduce the overall execution time.

III. C. Preconditioner 2

A core-wise grouping of the unknown types is used in the second type of preconditioner. This grouping allows us to associate parts of the Jacobian to the underlying balance equations since they are distinctly separated from one another. The preconditioner can be symbolically written as

$$\bar{\bar{P}}_2 = \begin{bmatrix} \bar{\bar{J}}_{11} & \bar{\bar{J}}_{12} & \bar{\bar{J}}_{13} \\ 0 & \bar{\bar{J}}_{22} & 0 \\ \bar{\bar{J}}_{31} & \bar{\bar{J}}_{32} & 0 \end{bmatrix}. \tag{30}$$

In order to apply the Porsching's algorithm, this preconditioner can be written in the same manner as Eq. (29), where by inspection we can write

$$\bar{\bar{\hat{J}}}_{11} = \begin{bmatrix} \bar{\bar{J}}_{11} & \bar{\bar{J}}_{12} \\ 0 & \bar{\bar{J}}_{22} \end{bmatrix}, \tag{31}$$

$$\bar{\bar{\hat{J}}}_{12} = \begin{bmatrix} \bar{\bar{J}}_{13} \\ 0 \end{bmatrix}, \tag{32}$$

and

$$\bar{\bar{\hat{J}}}_{21} = \left[\bar{\bar{J}}_{31} \; \bar{\bar{J}}_{32}\right]. \tag{33}$$

When calculating the action of $\bar{\bar{\hat{J}}}_{11}^{-1}$, the BILU approximation is used to determine the action of $\bar{\bar{J}}_{11}^{-1}$. In addition, since each diagonal block of the $\bar{\bar{J}}_{22}$ submatrix is a lower triangular matrix associated with coolant flow up a coolant channel, the action of $\bar{\bar{J}}_{22}^{-1}$ can be completed by performing a forward sweep.

This preconditioner was formulated to reduce the cost of factorizing the $\bar{\bar{\hat{J}}}_{11}$ block and subsequently the cost of performing the BILU approximation of Preconditioner 1. While the same block seven banded structure is possessed by both the $\bar{\bar{\hat{J}}}_{11}$ block of Preconditioner 1 and the $\bar{\bar{J}}_{11}$ block of Preconditioner 2, the size of each block entry is significantly reduced from 6x6 in Preconditioner 1 to just 2x2 in Preconditioner 2. However, while in Preconditioner 1 all entries of the Jacobian matrix are used in the preconditioner, in Preconditioner 2 the $\bar{\bar{J}}_{21}$ block, which physically represents the changes in the thermal hydraulics properties as a result of the change in flux (power) distribution, is eliminated. The rest of the Krylov algorithm, where all entries of the Jacobian matrix are considered, will correct for this imperfection.

III. D. Preconditioner 3

Preconditioner 3 also employs the core-wise grouping of the unknown types. However, the original Jacobian matrix is slightly rearranged in this preconditioner. The preconditioner can be symbolically written as

$$\bar{\bar{P}}_3 = \begin{bmatrix} \bar{\bar{\tilde{J}}}_{11} & \bar{\bar{0}} \\ \bar{\bar{\tilde{J}}}_{21} & \bar{\bar{\tilde{J}}}_{22} \end{bmatrix}, \tag{34}$$

where

$$\bar{\bar{\tilde{J}}}_{11} = \begin{bmatrix} \bar{\bar{J}}_{11} & \bar{\bar{J}}_{13} \\ \bar{\bar{J}}_{31} & 0 \end{bmatrix}, \tag{35}$$

$$\bar{\bar{\tilde{J}}}_{21} = \begin{bmatrix} \bar{\bar{J}}_{21} & 0 \end{bmatrix}, \tag{36}$$

and

$$\bar{\bar{\tilde{J}}}_{22} = \begin{bmatrix} \bar{\bar{J}}_{22} \end{bmatrix}. \tag{37}$$

When solving the preconditioner matrix system, due to the structure of the matrix, changes in the flux distribution and eigenvalue can now be calculated first and subsequently used for determining the changes in the thermal hydraulics properties. This is similar to the fission source iterations and thermal hydraulics update in the traditional iterative solver. The Porsching's algorithm is used when determining the action of $\bar{\bar{\tilde{J}}}_{11}^{-1}$. The BILU approximation is used to determine the action of $\bar{\bar{J}}_{11}^{-1}$ within the Porsching's algorithm.

III. E. Performance of the Preconditioners

Numerical experiments have been performed to examine the performance of these preconditioners. A preliminary study was completed using three Krylov methods, namely BICGSTAB, CGS, and restart GMRES methods. The purpose of this preliminary study is to select a Krylov method to be employed in the final implementation of the Newton-Krylov solver

and to observe the performance of the preconditioners. The chosen Krylov method will be examined further to check whether the conclusion from this preliminary study still holds.

For the preliminary study, a tight stopping criteria is used for the Krylov solvers to avoid Krylov iteration error contamination of the Newton iteration. As a result, the same number of Newton iterations is observed for all cases, enabling us to fairly compare the number of Krylov iterations from these cases. The results from this study are summarized in Tables III and IV. The stopping criteria for the Krylov iterations is $\varepsilon_{Krylov} = 5\times10^{-5}$ and the stopping criteria for the Newton iteration is $\varepsilon_{true} = 1\times10^{-3}$. The true error is defined as

$$\varepsilon_{true} = \left\| \frac{\bar{\phi}^{(m)} - \bar{\phi}^{(ref)}}{\bar{\phi}^{(ref)}} \right\|_2 = \sqrt{\frac{1}{2N} \sum_{n=1}^{N} \left\{ \left[\frac{\phi_{1,n}^{(m)} - \phi_{1,n}^{(ref)}}{\phi_{1,n}^{(ref)}} \right]^2 + \left[\frac{\phi_{2,n}^{(m)} - \phi_{2,n}^{(ref)}}{\phi_{2,n}^{(ref)}} \right]^2 \right\}}, \tag{38}$$

where N is the total number of spatial fuel nodes and m represents the current Newton iteration count. The reference flux solution is obtained using the traditional iterative algorithm with a set of extremely tight stopping criteria. Also note that the preliminary study was performed on a somewhat simplified 368 assembly BWR/4 core model. Although the geometry of the core is maintained, the core is loaded with only three types of fuel bundles and the core does not have radial reflectors. In addition, since the spacer-void model is not included in the derivation of the Jacobian matrix, algebraic variable reduction allows us to reduce the number of thermal hydraulics unknowns included in Newton's method treatment to three, namely the node-wise moderator pressure, density, and internal energy.

Four things were learned from this study. Firstly, from Table III one can see that Preconditioner 1, which carries the most information from the original Jacobian matrix, needs the fewest Krylov iterations to satisfy the stopping criteria with respect to the other two

preconditioners. Unfortunately, the computational cost of BILU factorizations for this preconditioner is much higher than for the other preconditioners, making the overall performance of this preconditioner inferior. Secondly, Table IV shows that while the other computational components of the Newton part of the solver stay relatively constant, the computational cost of the Jacobian matrix factorization is reduced by a factor of four for Preconditioners 2 and 3 relative to Preconditioner 1. Subsequently, when applying the preconditioner in the Krylov part of the solver, the computational costs for the solution of the preconditioned matrix system for Preconditioners 2 and 3 are also reduced relative to the computational cost for Preconditioner 1. For the preconditioner performance part of the study, we conclude that Preconditioner 3 provides the lowest total solution time. Finally, for the Krylov solver part of the study, observing the results shown in Table III, one can see that the BICGSTAB method outperforms the other Krylov methods and, therefore, will be employed in the final implementation of the Newton-Krylov solver along with Preconditioner 3.

Additional studies were performed to examine the performance of the preconditioners. For this study, a more realistic 368 assembly GE BWR/4 core is utilized. Also note that for this study the spacer-void model has been explicitly included as a part of the Jacobian matrix system. This modification adds the void fraction as an additional unknown. While this change does not impact the LU factorization cost for Preconditioners 2 and 3 since the submatrix to be factorized for these preconditioners only has the neutronics part of the problem, *i.e.* the derivative of the neutron diffusion equation, it changes the LU factorization cost of Preconditioner 1 dramatically since now each entry of the $\hat{\bar{\bar{J}}}_{11}$ becomes a 6x6 versus 5x5 matrix.

The results from this study are presented in Table V. While the cost for performing other parts of the Newton iterations remains unchanged, the cost of the Jacobian matrix factorization for

Preconditioners 2 and 3 is now reduced more significantly, by a factor of ~20, with respect to Preconditioner 1. Subsequently, compared to Preconditioner 1, the cost of the preconditioner solve is reduced by 45% and 70% for Preconditioners 2 and 3, respectively. Regardless of the BICGSTAB stopping criteria, the results presented in this table exhibit a consistent trend with the results from the preliminary study; hence, we will utilize the BICGSTAB solver along with Preconditioner 3 in the final implementation of the Newton-Krylov solver.

IV. IMPLEMENTATION OF INEXACT NEWTON'S METHODS

Consider Eq. (8), a non-linear system of equations

$$\bar{\Gamma}(\bar{x}) \equiv (\bar{B}(\bar{\Theta}) - \bar{Q} = \bar{0}) . \tag{39}$$

In the implementation of the exact Newton's method, the Jacobian ($\overline{\overline{J_\Gamma}}$) of the $\bar{\Gamma}$ operator is first determined. If necessary, the Jacobian matrix is factorized (actually the preconditioner matrix which approximates $\overline{\overline{J_\Gamma}}$) so that the results of the factorization can later be used as part of the algorithm for solving the linearized system of equations. Regardless of the solver used, the following linearized system of equations needs to be solved to obtain the next iterative values of the unknowns ($\bar{\Theta}^{(m+1)}$):

$$\bar{\Theta}^{(m+1)} = \bar{\Theta}^{(m)} - \overline{\overline{J_\Gamma}}\Big|_{\bar{\Theta}^{(m)}} \bar{\Gamma}(\bar{\Theta}^{(m)}) , \tag{40}$$

where m is the Newton iterative count and $\overline{\overline{J_\Gamma}}\Big|_{\bar{\Theta}^{(m)}}$ is the Jacobian of $\bar{\Gamma}$ evaluated at the current iterative value of the unknowns ($\bar{\Theta}^{(m)}$). The algorithm for the overall Newton's method is shown in Figure 5.

IV. A. The Chord Method

The costs for computing the Jacobian coefficient matrix and its factorization are significant. One approach to reduce this cost is to move these operations outside the loop indicated on Figure 5. This implies that the linear approximation of $\bar{\Gamma}(\Theta) = \bar{0}$ solved at each Newton iteration has the derivative (Jacobian coefficient matrix) determined at only the initial iteration. This method is called the Chord method [8] and the algorithm for this method is shown in Figure 6. For the Chord method, the linearized system of equations solved at each iteration can be symbolically written as

$$\bar{\Theta}^{(m+1)} = \bar{\Theta}^{(m)} - \bar{\bar{J}}_{\Gamma}\Big|_{\bar{\Theta}^{(0)}} \bar{\Gamma}(\bar{\Theta}^{(m)}). \tag{41}$$

Although the cost per Newton iteration can be considerably reduced by utilizing the Chord method, there is a trade-off between this reduction and the increase in the total number of Newton iterations due to the approximation introduced by this method. Therefore, a successful execution of a Chord method will highly depend upon the efficiency of the linearized equations solver. The Chord method also has a higher likelihood of not converging due to misdirection of the search.

IV. B. The Shamanskii's Method

The Shamanskii's method [8] is introduced to help alleviate the excessive increase in the Newton iterations resulting from utilizing the Chord method by providing a better approximation of the Jacobian coefficient matrix. This is accomplished by updating the Jacobian coefficient matrix at a certain frequency. The algorithm for this method is shown in Figure 7. For the Shamanskii's method, the linearized system of equations solved at each iteration can be written as

$$\bar{\Theta}^{(m+1)} = \bar{\Theta}^{(m)} - \bar{\bar{J}}_{\Gamma}\Big|_{\bar{\Theta}^{(m(m))}} \bar{\Gamma}(\bar{\Theta}^{(m)}), \tag{42}$$

where $\bar{m}(m)$ indicates the past Newton iteration during which the Jacobian matrix is updated for the m^{th} Newton iteration. The number of Newton iterations for the Shamanskii's method should fall between those of exact Newton and Chord methods.

In utilizing the Shamanskii's method, a parametric study should be performed to determine the optimum frequency for the Jacobian matrix update. An update frequency of N denotes that the Jacobian matrix is updated every N Newton iterations. Therefore, using an update frequency of 1 makes the Shamanskii's method equivalent to the exact Newton's method and using an update frequency of ∞ makes it equivalent to the Chord method.

V. RESULTS

Presented in this section are the results from testing the Newton-BICGSTAB solver utilizing a single loading pattern (LP) as well as multiple LPs. The result from a control rod pattern optimization run is also included to show the performance of the solver for an optimization run. We use this approach to test the robustness of the solver against an increasing number of problems starting from analyzing a single configuration, *e.g.* when performing preconditioner studies, to evaluating numerous core configurations during an optimization run. We concluded this section with a brief discussion on the performance of inexact Newton's methods.

V. A. Evaluation of a Single Loading Pattern

After deciding on utilizing the preconditioned BICGSTAB (with Preconditioner 3) as the solver for the linearized system of equations, we tested the solver on a single loading pattern to observe the average behavior of the solver over several burnup steps. This exercise is also

intended to find a set of stopping criteria which provides the highest speedup while still maintaining the robustness of the solver.

The model used for this study is an 800 assembly GE BWR/6 core. The cycle modeled has 17 burnup steps. The nodal method is employed to solved the three-dimensional, two-group neutron diffusion equation. The three-equation mixture drift flux model is used to treat the two-phase flow in the core. An inlet flow redistribution calculation is also performed to obtain uniform inlet and exit pressures across the core. In addition, the critical flow search is also executed.

The test is actually divided into two parts. In the first part, the true errors in the neutron flux distribution and the moderator density distributions are used to terminate the overall Newton-BICGSTAB solver. The true error for the neutron flux distribution is defined as follow

$$\varepsilon_{true}^{\phi} = \left\| \frac{\bar{\phi}^{(m)} - \bar{\phi}^{(ref)}}{\bar{\phi}^{(ref)}} \right\|_2 = \sqrt{\frac{1}{2N} \sum_{n=1}^{N} \left\{ \left[\frac{\phi_{1,n}^{(m)} - \phi_{1,n}^{(ref)}}{\phi_{1,n}^{(ref)}} \right]^2 + \left[\frac{\phi_{2,n}^{(m)} - \phi_{2,n}^{(ref)}}{\phi_{2,n}^{(ref)}} \right]^2 \right\}}, \tag{43}$$

where N denotes the total number of spatial fuel nodes and m represents the current Newton iteration count. Similarly, the true error for the moderator density distribution is defined as

$$\varepsilon_{true}^{\rho} = \left\| \frac{\bar{\rho}^{(m)} - \bar{\rho}^{(ref)}}{\bar{\rho}^{(ref)}} \right\|_2 = \sqrt{\frac{1}{N} \sum_{n=1}^{N} \left[\frac{\rho_n^{(m)} - \rho_n^{(ref)}}{\rho_n^{(ref)}} \right]^2}. \tag{44}$$

The reference quantities are obtained using the traditional iterative algorithm with a set of extremely tight stopping criteria. The results from this part of the test are presented in Table VI. Speedup in Table VI denotes the ratio of the CPU times for the traditional solver to Newton-BICGSTAB solver, both utilizing the same true error stopping criteria. We have chosen the true stopping criteria to be 1×10^{-2} for both the neutron flux and moderator density distributions. This implies that the results from these solvers are, on average, within 1% of the true solutions. This

test shows that the Newton-BICGSTAB solver can provide a speedup of 1.67 with respect to the traditional solver. However, in practice the true solutions are not available and, hence, the measure of true error cannot be utilized as the stopping criteria. Therefore, we still need to evaluate the performance of the solver when the real stopping criteria are utilized.

The second part of the test is aimed to answer this question. The stopping criteria utilized for the overall solver are 1e-4, 5e-4, and 5e-2 for ε_k, ε_F, and ε_ρ (as defined in Eqs. (24) through (26)), respectively. The results from this part of the test are summarized in Table VII and Figure 8. Speedup in Table VII again denotes the ratio of the CPU times for the traditional solver to Newton-BICGSTAB solver, both utilizing the same overall stopping criteria. The purpose of examining multiple pairs of Newton and Krylov stopping criteria is three-fold. First, we would like to observe the variation in the number of Newton iterations as a function of stopping criteria. Figure 8 shows that for a particular set of stopping criteria, the number of Newton iterations does not change much throughout the depletion steps. Furthermore, as expected the total number of Newton iterations increases as the stopping criteria is reduced. Second, we would like to understand the trade-off between how much we gain in terms of improved agreement in the results against how much we lose in terms of potential speedup. Results in Table VII indicate that while there is no apparent improvement in the quality of the results as tighter stopping criteria are employed, significant decreases in the speedup factor are observed. These results suggest that a set of relatively relaxed stopping criteria should be employed by the solver. Lastly, we would like to examine the possibility of employing a set of dynamic stopping criteria, *i.e.* changing stopping criteria as the iteration progresses. This is what Case 7 in Table VII represents. The philosophy used in dynamically changing the stopping criteria is that a more relaxed stopping criteria can be employed early in the iteration, while a tighter stopping criteria is needed when we are close to a

converged solution. For example, employing a relaxed BICGSTAB stopping criteria in the first couple of Newton iterations will surely introduce error contaminations to the Newton iteration. However, at this stage since we are still far away from the solution, so we believe that these contaminations are much less significant than the error introduced by the first-order approximation of the Newton's method. Later on when we are close to the solution, since the Jacobian matrix is now "pointing" in the right direction and we would like to reach the solution as fast as possible, ideally in one iteration, we have to solve for the changes in the variables as accurately as we can; therefore, a tighter BICGSTAB stopping criteria is employed. The result shown in the last column of Table VII indicates that this approach can produce higher speedup than the statically determined set of stopping criteria.

V. B. Evaluation of Multiple Loading Patterns

The robustness of the solver utilizing the chosen set of stopping criteria from the single LP study, *i.e.* the dynamically determined stopping criteria, is examined further by testing the solver using multiple LPs. Two core types are evaluated during this study. The first core is an 800 assembly GE BWR/6 core. The cycle modeled has 17 burnup steps. Two-phase thermal hydraulics model with inlet flow redistribution is employed. In addition, the critical flow search is also performed. The second core is a 368 assembly GE BWR/4 core. The cycle has 10 depletion steps. The same thermal hydraulics model as the first core type is employed; however, no critical flow search is completed. For both core types, the nodal method is applied for solving the three-dimensional, two-group neutron diffusion equation.

The results from the multiple loading patterns evaluation are presented in Figures 9 and 10 for the first and second core types, respectively. The top part of Figure 9 depicts the agreement in the end of cycle (EOC) flow fraction between the traditional and the Newton-BICGSTAB solvers

for the first core type. The stopping criteria for the critical flow search, *i.e.* the maximum difference allowed between the predicted and target k_{eff} value, is set to 25 pcm for this problem. Therefore, the difference between the two converged k_{eff} values can be as large as 50 pcm, which clearly can deteriorate the agreement in the EOC flow fraction as well. The top part of Figure 10 presents the agreement in the k_{eff} value between the two solvers for the second core type. The bottom part of these two figures show the agreement in the core limiting thermal margins, *e.g.* MFLPD, MAPRAT, and MFLCPR. Based on these plots, one can see that the agreement between the two solvers is excellent. The average speedups for the 25 LPs evaluated are 1.708 and 1.697 for the 800 and 368 assembly cores, respectively, consistent with the result from the single LP evaluation. On average, we can attribute approximately 48% of the speedup to the BICGSTAB solver and the remaining 52% to the Newton's method.

V. C. Results from a Control Rod Pattern Optimization Run

Since the principal objective of this project is to reduce the CPU execution time of an optimization run, it is appropriate to use an optimization run as the ultimate robustness test for the Newton-BICGSTAB solver. The core model used for the optimization is an 800 assembly GE BWR/6 core. For the thermal hydraulics model, in addition to the model used in the previous studies, the spacer-void model which corrects the void fraction for the effects of spacer grids is also incorporated. The cycle analyzed has 10 depletion steps.

The results from this study are summarized in Table VIII. Both optimization runs satisfy the thermal margin constraints imposed and produce similar quality results. With regard to the overall execution time, the Newton-BICGSTAB solver provides a speedup of 1.896. However, after taking into account that a different number of CRPs are evaluated in each run, the average

speedup gained by the new solver is around 1.61, which is consistent with what was observed in the previous studies.

V. D. Results from the Inexact Newton's Method Study

The feasibility of employing an inexact Newton's method to provide an additional speedup to the Newton-BICGSTAB solver has been assessed. The study was performed on the same core model employed in the single LP evaluation above. The results from this study are presented in Table IX. From this table, one can see that initially a less frequent update of the Jacobian matrix, *i.e.* higher n value in the Shamanskii[n] method, provides additional speedup. This follows since the CPU time savings associated with less frequent Jacobian matrix updates more than offsets the penalty of increasing number of Newton iterations. However, if the frequency of the Jacobian matrix update is relaxed further, speedup is degraded since the increase in Newton iterations more than offsets the CPU time savings associated with less frequent Jacobian matrix updates. This trade-off results in an optimum performance by the Shamanskii[5]-BICGSTAB solver, which provides an additional 10% reduction in the execution time.

When implementing the Shamanskii's method, we expected that the number of Newton iterations will grow with respect to the exact Newton's method due to the inexact nature of the method, but should not exceed the number of Newton iterations associated with the Chord method. Table IX exhibits this expected behavior, where the numbers of Newton iterations associated with the exact Newton's method and the Chord method provide the lower and upper limits for the Shamanskii's method.

VI. CONCLUSIONS AND AREAS FOR FUTURE PROJECTS

The main objective of this project, which is to reduce the CPU execution time of BWR core simulators, has been accomplished by virtue of the implementation of a robust and computationally efficient Newton-BICGSTAB solver. An overall speedup of 1.7 has been achieved with reference to the traditional solver. If speedup is only measured for the solver portion of the core simulator, the speedup increases to 1.9. In terms of what achieves the overall speedup, about 48% is associated with BICGSTAB and the remaining 52% is associated with Newton's method. The robustness of the solver has been tested against numerous core configurations and consistent results have been observed each time. More importantly, this project has laid a foundation upon which numerous future projects can be developed in this new area of nuclear engineering research.

A key to a successful implementation of the new solver is the development of a preconditioner for the BICGSTAB solver. The performance of the current Newton-BICGSTAB solver can likely be further improved by devising a more sophisticated preconditioner. For example, a preconditioner which is suitable for parallel computations could substantially further reduce the computational time.

The implementation of an appropriate inexact Newton's method is another way to improve the performance of the Newton-BICGSTAB solver. However, the implementation is not limited to the Chord and Shamanskii's method, but also includes any approximation introduced to the exact Newton's method. For example, keeping other parts of the exact Newton-BICGSTAB algorithm unchanged, we can elect to perform the LU factorization of the preconditioner matrix less frequently to save execution time.

Inclusion of more non-linear variables, *e.g.* the weak feedbacks, in the Newton's method treatment is another topic worthy of examination. This will require the development of a new preconditioner since the structure of the Jacobian matrix changes as more variables are included.

VII. REFERENCES

[1] R. Barrett, M. Berry, T. F. Chan, J. Demmel, J. Donato, J. Dongarra, V. Eijkhout, R. Pozo, C. Romine, and H. Van der Vorst, *Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods*, SIAM, Philadelphia (1993).

[2] T. Downar, J. Y. Wu, and R. Janardhan, "Parallel and Serial Applications of the RETRAN-03 Power Plant Simulation Code Using Domain Decomposition and Krylov Subspace Methods," *Nucl. Tech.*, **117**, pp. 133-150 (1997).

[3] R. W. Freund and N. M. Nachtigal, "QMR: A Quasi-Minimal Residual Method for Non-Hermitian Linear System," *Numer. Math. Journal*, Vol. 60 (1991).

[4] R. Janardhan and T. Downar, "A Nested FGMRES Method for Parallel Calculation of Nuclear Reactor Transients," *Journal of Scientific Computing*, Vol. 13, No. 1, pp. 65-93 (1998).

[5] H. G. Joo, *Domain Decomposition Methods for Nonlinear Nodal Spatial Kinetics*, Ph.D. Dissertation, Purdue University (1996).

[6] H. G. Joo and T. Downar, "Incomplete Domain Decomposition Preconditioning for Coarse Mesh Neutron Diffusion Problems," *Nucl. Sci. Eng.*, **123**, pp. 403-414 (1996).

[7] Atul Karve and Paul J. Turinsky, "FORMOSA-B: A Boiling Water Reactor Incore Fuel Management Optimization Package II," *Nucl. Tech.*, **131**, pp. 48-68 (2000).

[8] C. T. Kelley, *Iterative Methods for Linear and Nonlinear Equations*, SIAM, Philadelphia (1995).

[9] C. T. Kelley, *Iterative Methods for Optimization*, SIAM, Philadelphia (1999).

[10] Brian R. Moore, Paul J. Turinsky, and Atul A. Karve, “FORMOSA-B: A Boiling Water Reactor Incore Fuel Management Optimization Package,” *Nucl. Tech.*, **126**, pp. 153-169 (1999).

[11] T. A. Porsching, J. H. Murphy, and J. A. Redfield, “Stable Numerical Integration of Conservation Equations for Hydraulic Networks,” *Nucl. Sci. Eng.*, **43**, pp. 218-225 (1971).

[12] Y. Saad and M. H. Schultz, “GMRES: A Generalized Minimal Residual Algorithm for Solving Non-Symmetric Linear Systems,” *SIAM J. Sci. Stat. Comput.*, Vol. 7, No. 3, pp. 856-869 (1986).

[13] P. J. Turinsky, *et. al.*, “NESTLE: A Few-Group Neutron Diffusion Equation Solver Utilizing the Nodal Expansion Method for Eigenvalue, Adjoint, Fixed-Source Steady-State and Transient Problems,” EGG-NRE-11406 (1994).

[14] H. A. Van der Vorst, “Bi-CGSTAB: A Fast and Smoothly Converging Variant of Bi-CG for the Solution of Nonsymmetric Linear Systems,” *SIAM J. Sci. Stat. Comput.*, Vol. 13, No. 2, pp. 631-644 (1992).

[15] J. Wu, *The Application of Domain Decomposition and Krylov Subspace Methods to Reactor Safety Analysis*, Ph.D. Dissertation, Purdue University (1997).

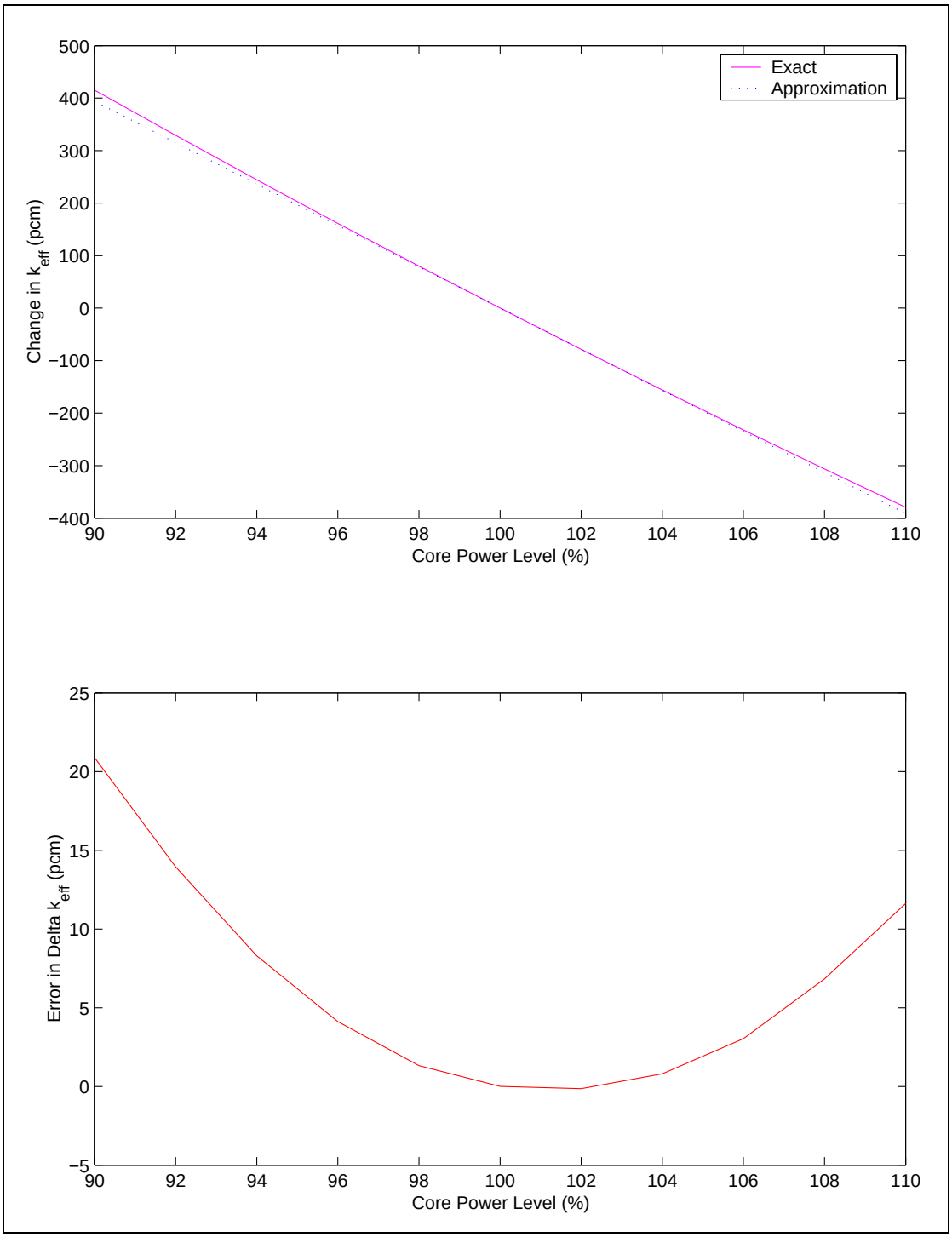


Figure 1. Linearization error of the k_{eff} value.

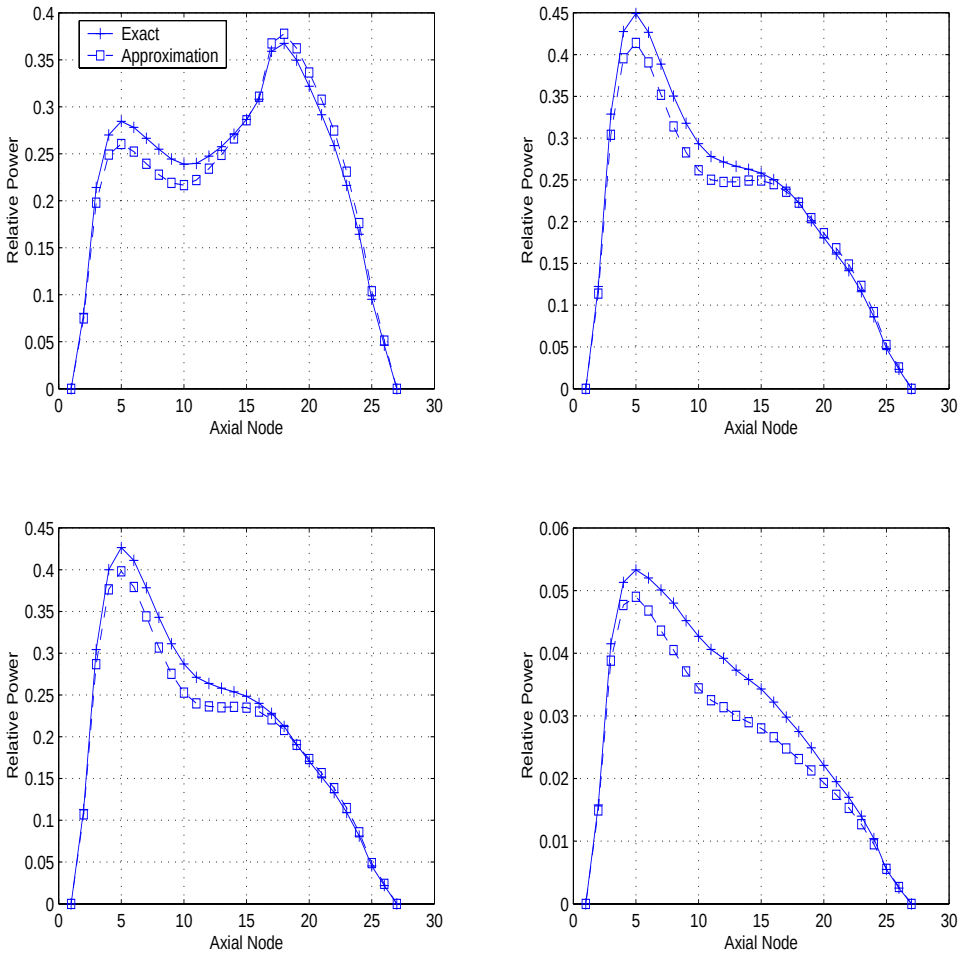


Figure 2. Exact vs. approximate fast flux changes for +10% power perturbation.

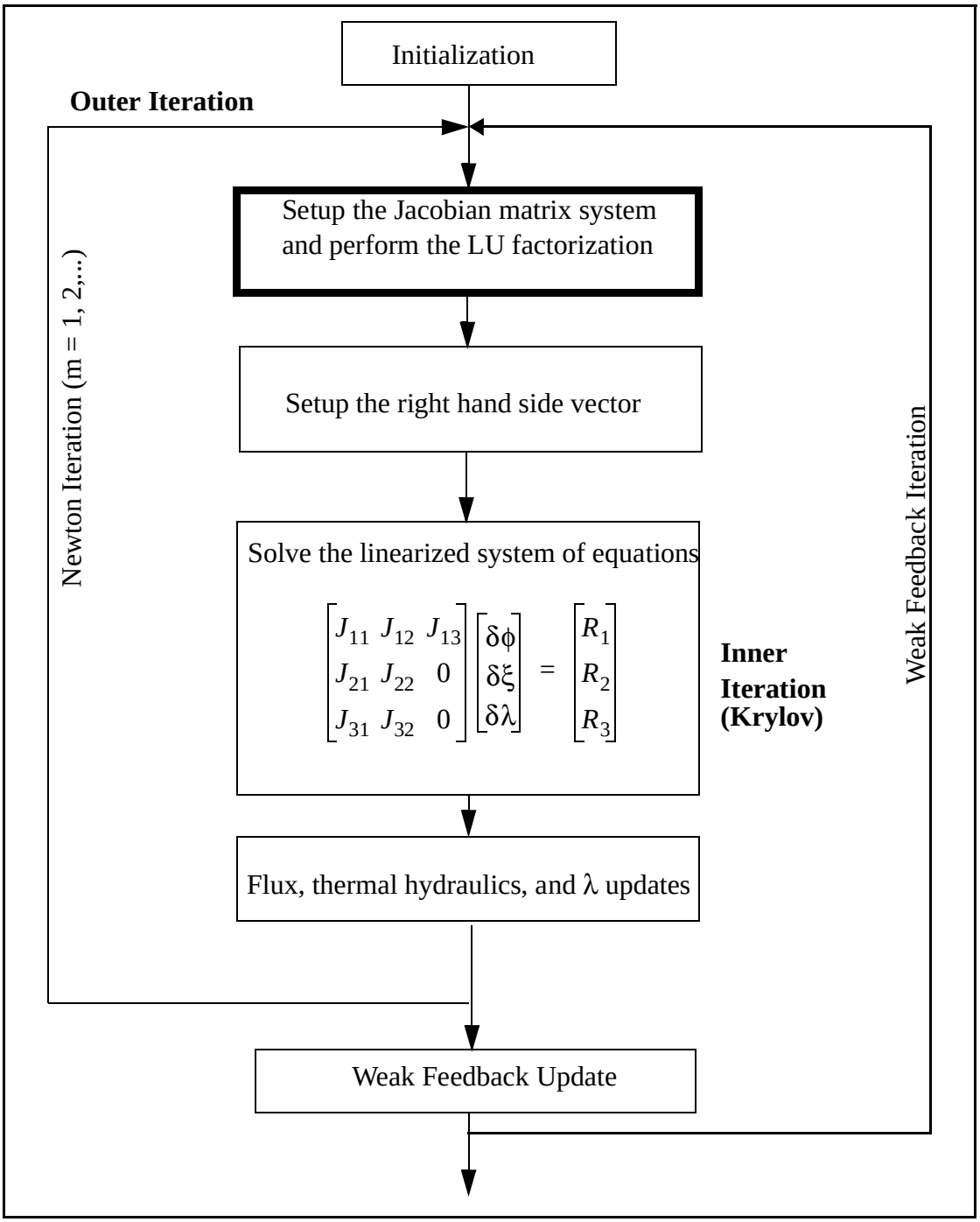


Figure 3: The Newton-Krylov algorithm.

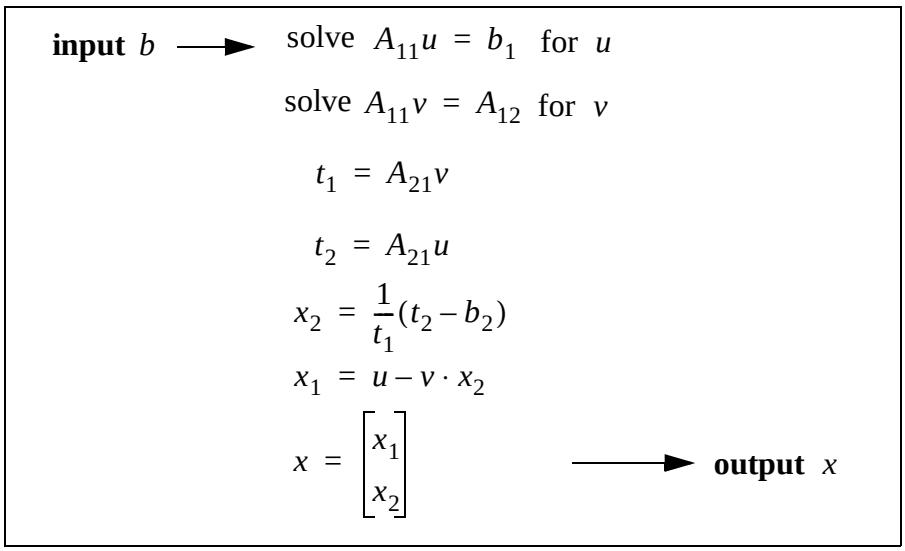


Figure 4: Porsching's algorithm.

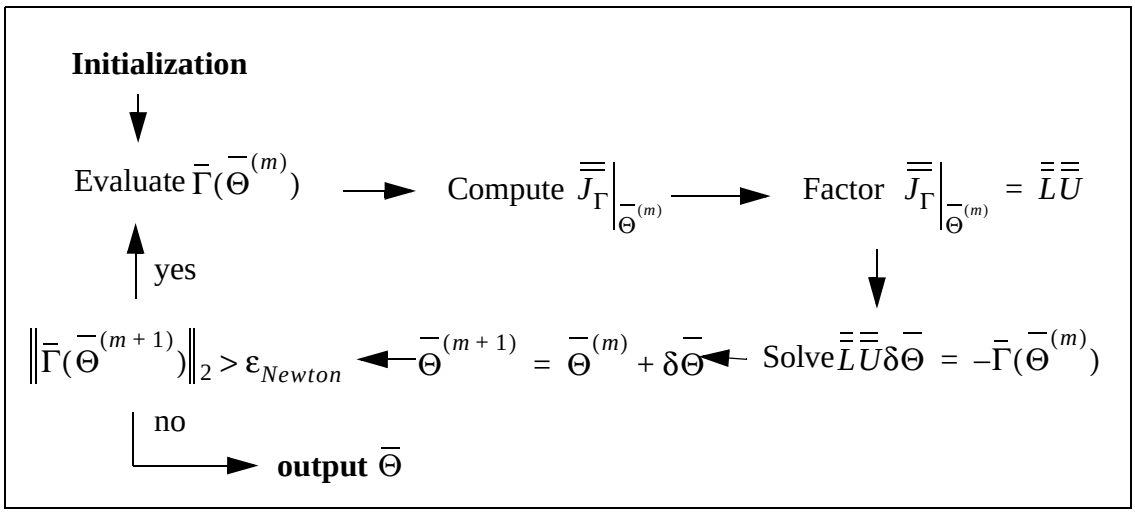


Figure 5. Exact Newton algorithm.

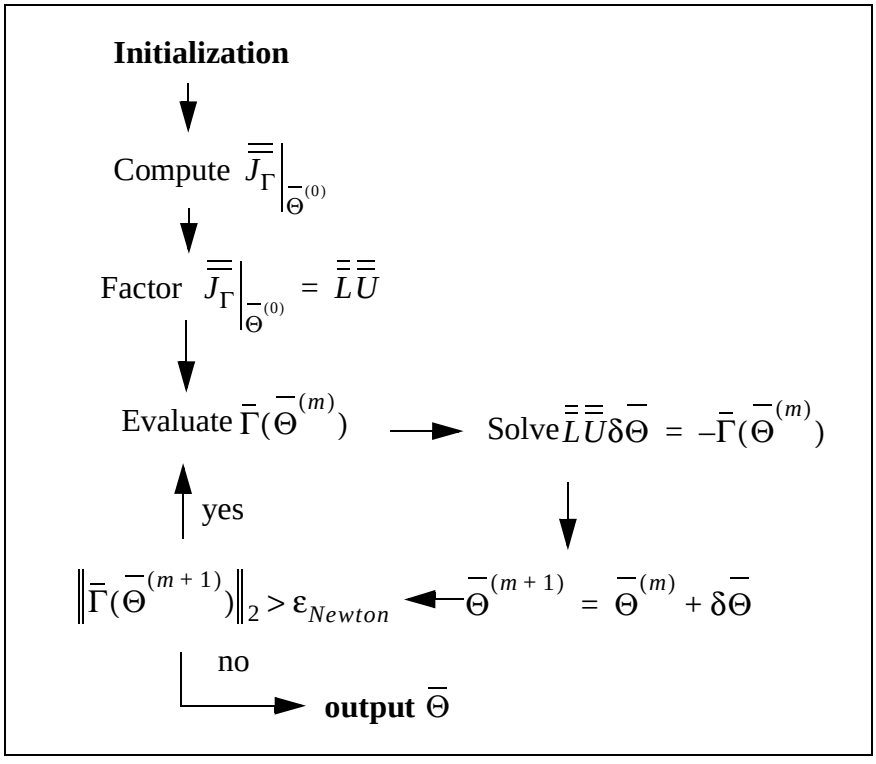


Figure 6. Chord method.

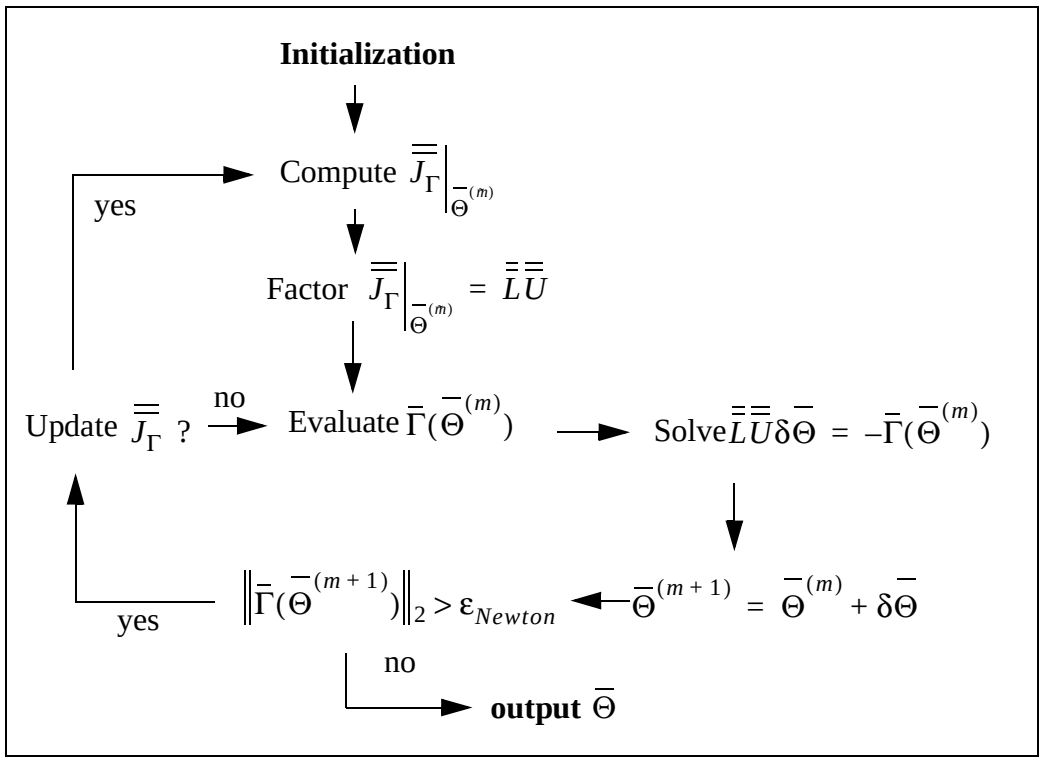


Figure 7. Shamanskii's method.

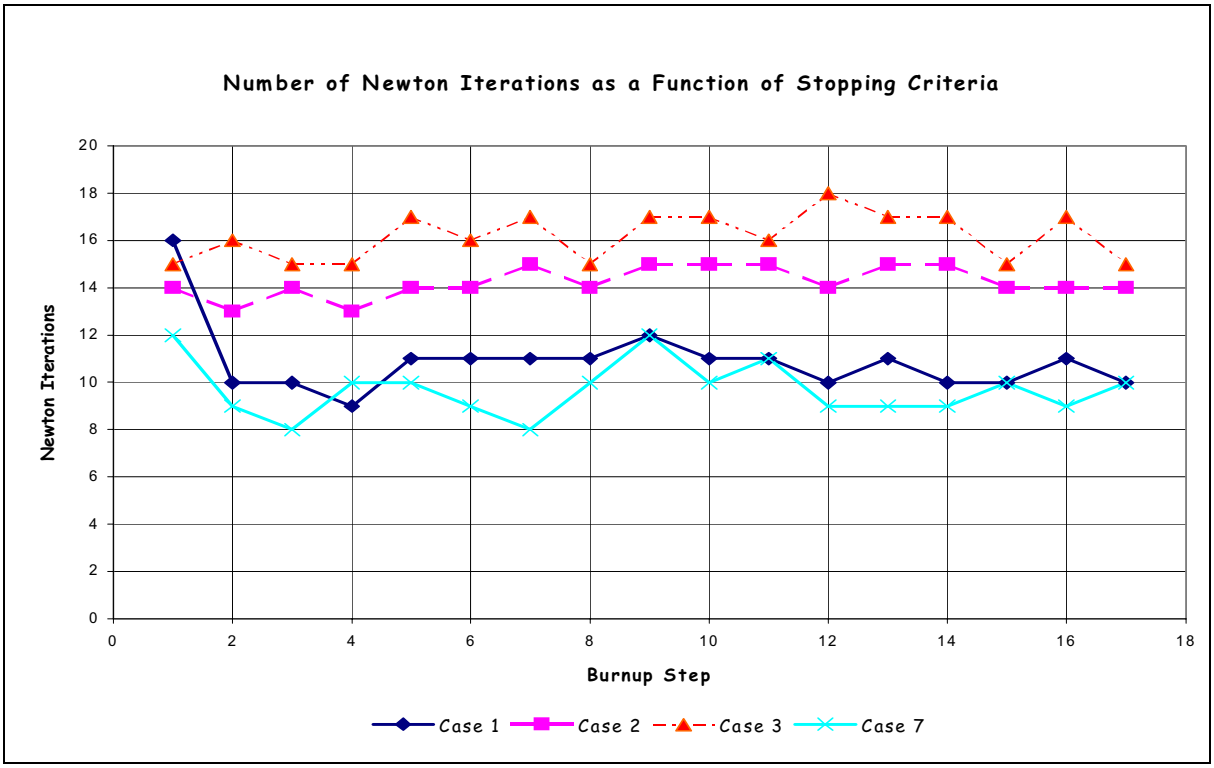
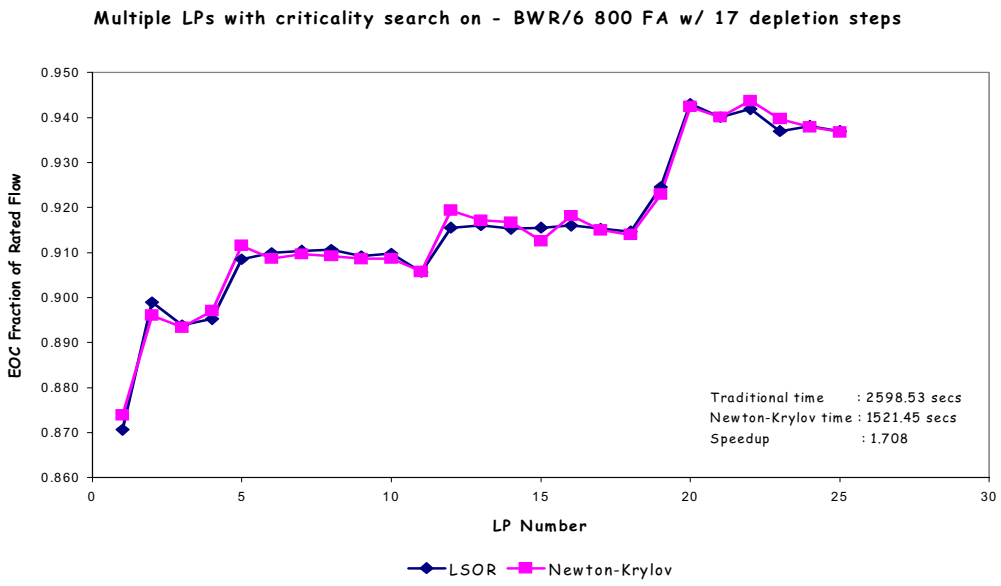


Figure 8. Newton iterations vs stopping criteria.



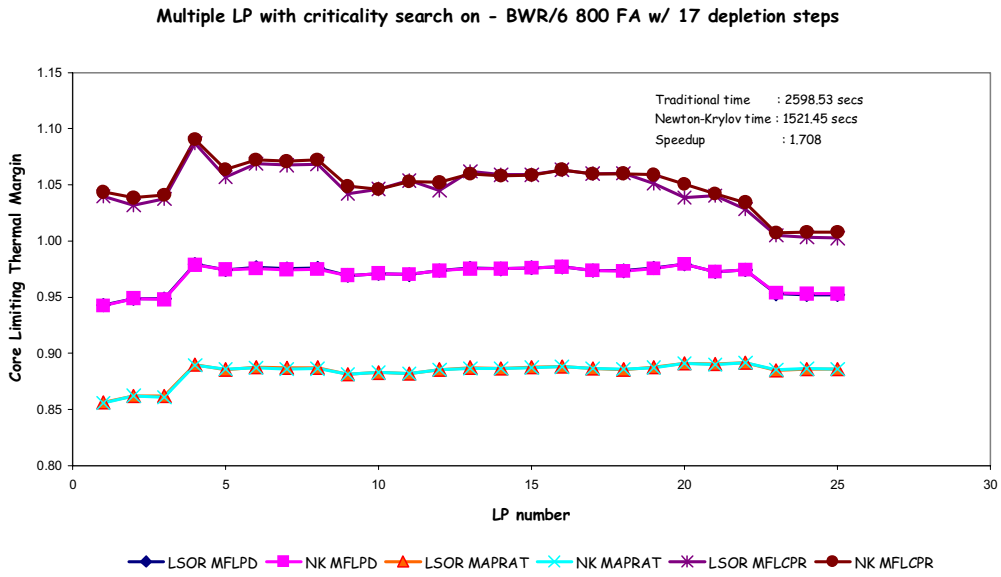
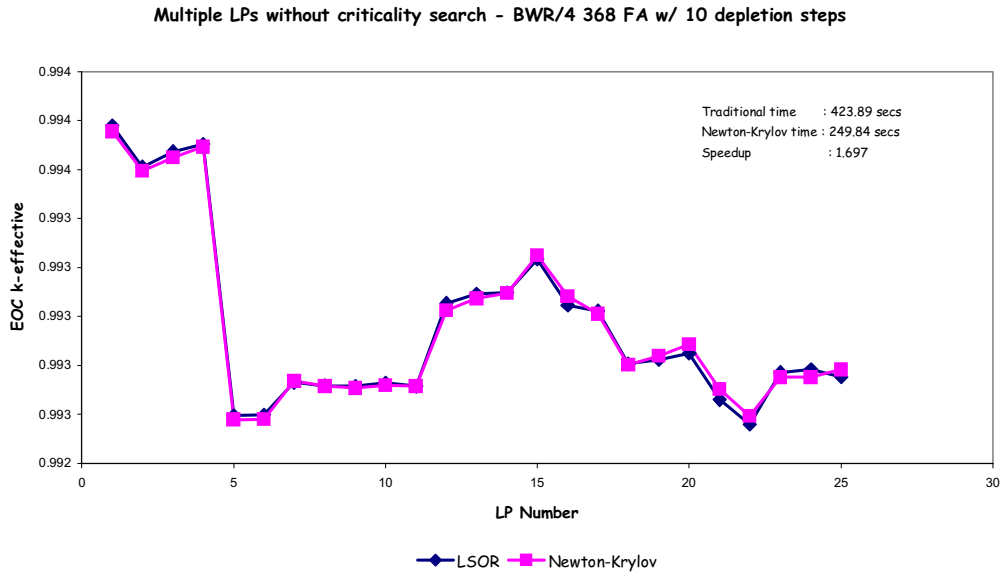


Figure 9. Multiple LPs results - 800 F/A GE BWR/6 core.



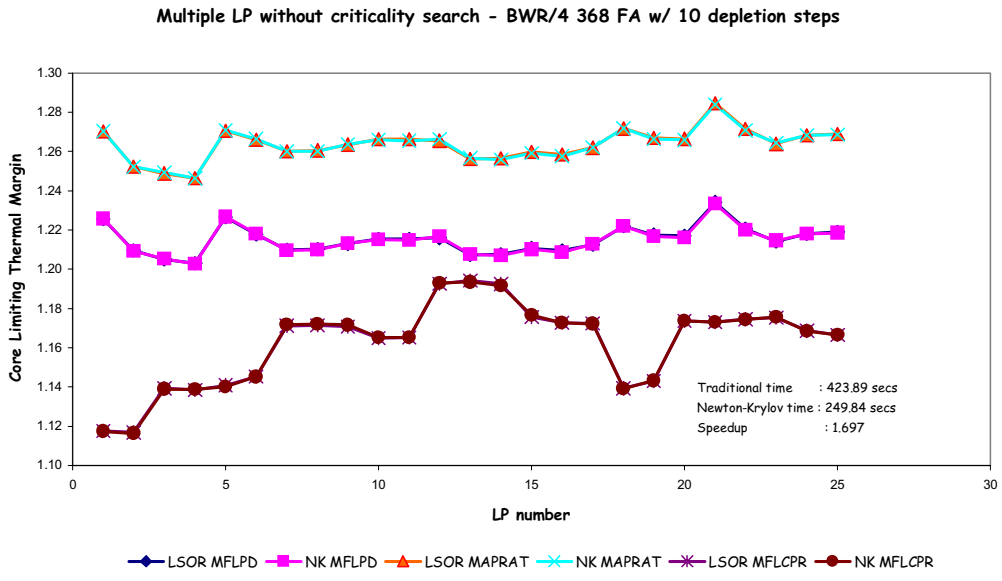


Figure 10. Multiple LPs results - 368 F/A GE BWR/4 core.

Table I: Cases Examined and Stopping Criteria.

Case	NEM	TH	Flow Redistribution	Critical Flow	Spacer Void	Fission Product
1	On	Off	Off	Off	Off	On
2	Off	On	Off	Off	Off	
3	Off	On	On	Off	Off	
4	Off	On	On	On	Off	
5	Off	On	Off	Off	On	
6	On	On	Off	Off	Off	
7	On	On	On	Off	Off	
8	On	On	On	On	Off	
9	On	On	On	On	On	
10	On	On	On	On	Off	Off

Stopping Criteria

$$\frac{\left|k_{eff}^{(l)} - k_{eff}^{(l-1)}\right|}{k_{eff}^{l}} \le \varepsilon_k ; \quad \frac{\left\|\rho^{(l)} - \rho^{(l-1)}\right\|_2}{\left\|\rho^{(l)}\right\|_2} \le \varepsilon_\rho ; \quad \frac{\left\|[\bar{\bar{F}}\bar{\phi}]^{(l)} - [\bar{\bar{F}}\bar{\phi}]^{(l-1)}\right\|_2}{\left\|[\bar{\bar{F}}\bar{\phi}]^{(l)}\right\|_2} \le \varepsilon_F$$

Note: l and $l-1$ are the two consecutive outer most loop iterations.

	ε_F	ε_k	ε_ρ
Loose Stopping Criteria	5e-4	1e-4	5e-2
Tighter Stopping Criteria	1e-4	5e-5	2e-4

Table II: Relative Changes in the Number of Outer Iterations.

Case	Core Type			
	368 assembly	560 assembly	724 assembly	800 assembly
Loose Stopping Criteria				
1	1.20	1.11	1.06	1.10
2	1.69	1.41	1.31	1.56
3	1.67	1.38	1.33	1.61
4	2.62	1.95	2.44	1.92
5	1.72	1.40	n/a	1.56
6	1.69	1.41	1.33	1.54
7	1.67	1.38	1.35	1.61
8	2.42	1.75	2.27	1.76
9	2.33	1.98	n/a	2.09
10	1.96	1.60	1.39	1.81
Tighter Stopping Criteria				
1	1.32	1.29	1.03	1.10
2	1.96	1.46	1.34	1.54
3	1.85	1.51	1.32	1.62
4	2.65	2.15	2.22	1.96
5	2.52	1.59	n/a	1.66
6	1.85	1.54	1.30	1.57
7	1.85	1.54	1.30	1.60
8	2.44	1.97	2.17	1.75
9	2.97	2.19	n/a	2.18
10	2.23	1.75	1.40	1.85

Table III: Preconditioner Performance for a Simplified 368 Assembly GE BWR/4 Core Model.

Variable	BICGSTAB			CGS			GMRES(5)		
	P_1	P_2	P_3	P_1	P_2	P_3	P_1	P_2	P_3
Krylov Iterations	44	47	51	57	61	63	93	130	127
Newton Iterations	5	5	5	5	5	5	5	5	5
Execution Time (in seconds[1])	3.88	3.15	2.97	4.47	3.60	3.28	5.42	5.35	4.84

1: Timing was performed on a 440 MHz SUN Ultra-10 machine.

Table IV: Detail Time Allocation (in seconds[1]) for a Simplified 368 Assembly GE BWR/4 Core Model.

Variable	BICGSTAB			CGS			GMRES(5)		
	P_1	P_2	P_3	P_1	P_2	P_3	P_1	P_2	P_3
Initialization	0.25	0.25	0.25	0.25	0.25	0.25	0.25	0.25	0.25
Newton part of the solver									
Jacobian Matrix Setup	0.42	0.42	0.43	0.42	0.42	0.42	0.42	0.42	0.43
Matrix Factorization	0.28	0.07	0.07	0.28	0.07	0.07	0.28	0.07	0.07
Variable Updates	0.21	0.21	0.21	0.21	0.21	0.21	0.21	0.21	0.21
Derivatives Calculation	0.38	0.38	0.37	0.37	0.37	0.39	0.37	0.37	0.38
Setup FDM Matrix and RHS Vector	0.10	0.10	0.10	0.10	0.10	0.10	0.10	0.10	0.10
Total Newton	1.39	1.18	1.18	1.38	1.17	1.19	1.38	1.17	1.19
Krylov part of the solver									
Preconditioner Solve	1.61	1.03	0.85	2.08	1.37	1.09	2.24	1.82	1.39
Matrix-Vector Operation	0.43	0.49	0.47	0.53	0.59	0.53	0.93	1.29	1.20
Inner Product	0.06	0.07	0.07	0.04	0.04	0.04	0.03	0.04	0.04
Vector Updates	0.10	0.10	0.11	0.15	0.14	0.14	0.51	0.70	0.69
Other Operations in Krylov	0.02	0.03	0.03	0.02	0.02	0.02	0.05	0.06	0.06
Total Krylov	2.22	1.72	1.52	2.82	2.16	1.82	3.76	3.91	3.38
Others	0.02	0.02	0.02	0.02	0.02	0.02	0.02	0.02	0.02

1: Timing was performed on a 440 MHz SUN Ultra-10 machine.

Table V: Preconditioner Performance (in seconds[1]) for a Realistic 368 Assembly GE BWR/4 Core Model.

Variable	$\varepsilon_{Krylov} = 1\times10^{-3}$			$\varepsilon_{Krylov} = 1\times10^{-4}$			$\varepsilon_{Krylov} = 1\times10^{-5}$		
	P_1	P_2	P_3	P_1	P_2	P_3	P_1	P_2	P_3
Initialization	0.048	0.048	0.048	0.048	0.048	0.049	0.049	0.048	0.048
Newton part of the solver									
Jacobian Matrix Setup	1.293	1.290	1.289	1.292	1.295	1.291	1.291	1.287	1.296
Jacobian Factorization	0.634	0.029	0.030	0.622	0.030	0.030	0.620	0.030	0.030
Variable Updates	0.514	0.514	0.514	0.513	0.514	0.514	0.513	0.514	0.515
Derivatives Calculation	0.042	0.042	0.043	0.043	0.042	0.043	0.043	0.043	0.043
Setup FDM Matrix and RHS Vector	0.088	0.088	0.088	0.088	0.088	0.088	0.088	0.088	0.087
Krylov part of the solver									
Preconditioner Solve	4.625	2.487	1.418	5.287	2.768	1.578	5.847	3.134	1.809
Matrix-Vector Operation	2.909	3.136	3.106	3.349	3.498	3.511	3.789	4.024	4.002
Inner Product	0.117	0.129	0.130	0.138	0.144	0.146	0.156	0.165	0.162
Vector Updates	0.217	0.237	0.233	0.250	0.267	0.262	0.281	0.310	0.297
Iterations									
Newton Iteration	8	8	8	8	8	8	8	8	8
Krylov Iteration	46	51	51	53	57	57	60	65	65
Avg. Krylov/Newton	5.75	6.38	6.38	6.63	7.13	7.13	7.50	8.13	8.13

1: Timing was performed on a 440 MHz SUN Ultra-10 machine.

Table VI: The Newton-BICGSTAB Performance on a Single LP Using True Error Stopping Criteria.

	Cases	
	Case 1	Case 2
Numerical Information		
Newton S.C.	1.00e-2	5.00e-3
Krylov S.C.	1.00e-2	5.00e-3
$\varepsilon_{true}^{\phi}$	1.00e-2	1.00e-2
$\varepsilon_{true}^{\rho}$	1.00e-2	1.00e-2
Newton/BU Step	11.53	12.94
Speedup	1.67	1.55
Results Comparison (average differences)		
k_{eff} (in pcm)	14.84	12.77
Flow Fraction	0.0041	0.0032
MFLPD	5.8235e-4	5.7059e-4
MAPRAT	6.8824e-4	6.2355e-4
MFLCPR	3.9882e-3	3.1765e-3

Table VII: The Newton-BICGSTAB Performance on a Single LP.

	Cases						
	Case1	Case2	Case3	Case4	Case5	Case6	Case7
Numerical Information							
Newton S.C.	1.00e-2	1.00e-3	5.00e-4	1.00e-2	1.00e-3	5.00e-4	Variable
Krylov S.C	1.00e-2	1.00e-2	1.00e-2	5.00e-3	5.00e-3	5.00e-3	Variable
Newton/BU Step	10.88	14.24	16.18	10.82	14.18	16.18	9.71
Speedup	1.59	1.29	1.17	1.59	1.31	1.18	1.76
Results Comparison (average differences)							
k_{eff} (in pcm)	15.91	16.39	17.31	15.49	18.14	17.17	15.46
Flow fraction	0.0044	0.0042	0.0044	0.0041	0.0045	0.0043	0.0042
MFLPD	6.00e-4	6.29e-4	6.65e-4	6.29e-4	7.06e-4	6.64e-4	5.71e-4
MAPRAT	7.82e-4	8.71e-4	8.94e-4	8.35e-4	9.29e-4	8.94e-4	7.94e-4
MFLCPR	4.25e-3	4.17e-3	4.35e-3	4.08e-3	4.52e-3	4.33e-3	4.01e-3

Table VIII: CRP Optimization Results.

Variable	Constraint	Reference Pattern	Optimized Pattern	
			LSOR	Newton-BICGSTAB
MFLPD	<0.940	0.9540	0.9417	0.9396
MAPRAT	<0.940	0.8598	0.8848	0.8733
MFLCPR	<0.940	0.9456	0.9298	0.9235
Number of CRPs evaluated			209	178
Time (sec[1])			1261	665

1: Timing was performed on a 2 GHz DELL 530 workstation.

Table IX: Inexact Newton's Method Results.

Method	Time[(1)] (seconds)	Newton Iteration	BICGSTAB Iteration	Weak Feedback Iteration
Newton	62.88	166	302	154
Chord	63.49	194	385	173
Shamanskii[2][(2)]	59.19	170	327	153
Shamanskii[3]	58.14	173	327	156
Shamanskii[4]	57.41	174	324	158
Shamanskii[5]	56.41	173	318	160
Shamanskii[6]	56.84	174	338	156
Shamanskii[7]	57.78	177	348	159
Shamanskii[8]	60.58	185	358	164

(1) : Timing was performed on a 2GHz DELL 530 workstation.
(2) : The n in Shamanskii[n] denotes the Jacobian matrix update frequency.